Model Checking Resource Bounded Systems with Shared Resources via Alternating Büchi Pushdown Systems

Nils Bulling[1] and Hoang Nga Nguyen[2]

[1] Delft University of Technology, The Netherlands
[2] School of Computer Science, University of Nottingham, UK

Abstract. It is well known that the verification of resource-constrained multi-agent systems is undecidable in general. In many such settings, resources are private to agents. In this paper, we investigate the model checking problem for a resource logic based on Alternating-Time Temporal Logic (ATL) with shared resources. Resources can be consumed and produced up to any amount. We show that the model checking problem is undecidable if two or more of such unbounded resources are available. Our main technical result is that in the case of a single shared resource, the problem becomes decidable. Although intuitive, the proof of decidability is non-trivial. We reduce model checking to a problem over alternating Büchi pushdown systems. An intermediate result connects to general automata-based verification: we show that model checking Computation Tree Logic (CTL) over (compact) alternating Büchi pushdown systems is decidable.

1 Introduction and Related Work

Research on resource-constrained multi-agent systems has become a popular topic in recent years, e.g. [7,8,2,13,1,3]. In particular, the verification of strategic agents acting under resource-constraints has been investigated by researchers; many of these approaches extend the alternating-time temporal logic (ATL) [4] with actions that, in the general case, consume or produce resources. If no bound on the possible amount of resources is given the model checking problems are easily undecidable [8]. Exceptions are possible if restrictions are imposed on the language [3] or on the semantics [1,13]. In many settings, resources are private to agents, each agent has its own set of resources. In [13] resources are shared and a resource money is used to claim resources. The authors present a decidable model checking result which is possible as the amount of resources is bounded. In this paper we are interested in the model checking problem where resources are shared and *unbounded*; resources can be consumed and produced without an upper bound on the total number of resources. The setting is rather natural. Resources are shared in e.g., the travel budget of a computer science department. All departmental members compete for the travel budget. Parts of the travel money of a successful grant application will be credited to the department's budget; there is no a priori bound on the total budget.

In this paper, we show that the model checking problem for the resource agent logic RAL [8] considered here is undecidable in general when there are more than two of such

unbounded shared resources. This result follows as a corollary from [8,3] where model checking resource bounded systems with private, unbounded resources has been proved undecidable. Secondly, we show that model checking RAL is decidable in case of a single shared, unbounded resource. Although this seems intuitive, as a single unbounded resource can intuitively be encoded by a single stack/counter, its proof is (technically) non-trivial and is based on a reduction to alternating Büchi pushdown systems [15,5]. We first introduce compact alternating Büchi pushdown systems (CABPDSs) to encode the resource bounded models of our logic such that the runs of the automaton can be related to execution trees of a given set of agents in the model. We show that model checking CTL over these systems is decidable using results of [15]. Finally, we reduce model checking RAL to model checking CTL over CABPDSs. These results extend work on model checking CTL over pushdown systems where atomic propositions can be given by regular languages [15]. The latter results, in turn, are based on [5] where reachability of alternating pushdown systems and model checking problems over pushdown systems with standard labelling functions are investigated. Model checking CTL over pushdown systems and its computational complexity have also been considered in [6]. Our model checking problem is also related to reachability in Büchi games [10]. Many complexity results about ABPDSs and their variants are known and established in the above mentioned pieces of work. In our future research we plan to determine the exact computational complexity of the model checking problem for resource agent logic (RAL) over 1-unbounded resource bounded models.

The paper is organised as follows. In Section 2 we discuss different resource types and introduce our version of resource agent logic with shared resources. In Section 3 we recall alternating Büchi pushdown systems (ABPDSs) and variants thereof. We propose compact ABPDSs for encoding our models. We show that model checking CTL over them is decidable. In Section 4 we give our main decidability result for a single unbounded resource. Finally, in Section 5 we consider the general case and show that model checking is undecidability if at least two unbounded resource types are available, and conclude in Section 6.

2 Resource Agent Logic

In this section we define the logic *resource agent logic* RAL and resource-bounded models. The framework is essentially based on [3]. We begin with a discussion on different resource types which can be classified among different dimensions:

Private resources are assigned to individual agents.
Shared resources can be accessed by all agents; they are global.
Consumable/producible resources can be consumed and produced. They often disappear after usage, like gasoline and energy, and may thus also be labelled *fluent*.
Re-usable resources do not in general disappear after usage. They may also be produced.
Bounded/unbounded resource types characterise whether arbitrarily many resources can be produced or if there is a bound on the maximal amount.

We note that the property of boundedness has a different flavour in comparison to the other properties. It is better understood as a property of the agents or of the specific

modelling rather than of the resource itself. For example, the agent can only carry up to two heavy boxes, or there are legislations which prohibit to have more than three cars in a household. In this paper we are interested in shared, unbounded, consumable resource types. That is, there is a common pool of resources for which all agents compete. Agents' actions may consume resources or produce them, always affecting the common pool of resources. Moreover, there can be arbitrarily many resources of a resource type.

Clearly, in real settings there will usually be a combination of different resource types. Adding bounded resources will in general not affect the decidability of model checking. Such resources can be encoded in the states, blowing up the model. One has to be more careful with unbounded resources. Having at least two unbounded resource types is often a first indication for an undecidable model checking problem if no other restrictions are imposed on the setting. Following the discussion above, we define a *(shared) endowment (function)* $\eta : \mathcal{R}es \to \mathbb{N}_0$ to specify the available shared resources of the resource types $\mathcal{R}es$ in the system; i.e., $\eta(r)$ is the number of shared resources of type r. With En we denote the set of all possible endowments. A special minimal endowment function is denoted by $\bar{0}$. It expresses that there are no resources at all.

Definition 1 (Shared resource structure, unbounded). *A* (shared) resource structure *is a tuple* $\mathfrak{R} = (\mathcal{R}es, \beta)$ *where* $\mathcal{R}es$ *is a finite set of shared consumable resources. Function* $\beta : \mathcal{R}es \to \mathbb{N} \cup \{\infty\}$ *is called* resource bound. *It specifies the maximal number of resources of a specific type in the model. We say that* $\mathfrak{R}$ *is* k-unbounded *iff the number of unbounded resource types is at most* k.

Syntax. Resource agent logic (RAL) is defined over a set of agents Agt and a set of propositional symbols Π. RAL-formulae[3] are essentially generated according to the grammar of ATL [4] as follows: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle^{\downarrow}\mathbf{X}\varphi \mid \langle\!\langle A \rangle\!\rangle^{\downarrow}\varphi\mathbf{U}\psi \mid \langle\!\langle A \rangle\!\rangle^{\downarrow}\mathbf{G}\varphi$ where $\mathsf{p} \in \Pi$ is a proposition and $A \subseteq \mathbb{A}\mathrm{gt}$ is a set of agents.

A formula $\langle\!\langle A \rangle\!\rangle^{\downarrow}\varphi$ is called *flat* if φ contains no cooperation modalities. The operators **X**, **U**, and **G** denote the standard temporal operators expressing that some property holds in the *next* point in time, *until* some other property holds, and *now and always* in the future, respectively. The *eventually* operator is defined as macro: $\mathbf{F}\varphi = \top\mathbf{U}\varphi$ (*now or sometime in the future*). The cooperation modality $\langle\!\langle A \rangle\!\rangle^{\downarrow}$ assumes that *all* agents in Agt act under resource constraints. The reading of $\langle\!\langle A \rangle\!\rangle^{\downarrow}\varphi$ is that *agents* A *have a strategy compatible with the currently available resources* to enforce φ. This means that the strategy can be executed given the agents' resources. Thus, it is necessary to keep track of resource production and consumption during the execution of a strategy.

Semantics. We define the models of RAL as in [3]. We also introduce a special class of these models in which agents have an *idle action* in their repertoire that neither consumes nor produces resources. Note that a model with idle actions is a special case of the general model.

Definition 2 (RBM, iRBM, unbounded). *A resource-bounded model (*RBM*) is given by* $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, \mathfrak{R}, t)$ *where* $\mathfrak{R} = (\mathcal{R}es, \beta)$ *is a shared resource*

[3] Note that we slightly change the notation in comparison with [8] where $\langle\!\langle A \rangle\!\rangle^{\downarrow}$ has the meaning of $\langle\!\langle A \rangle\!\rangle^{\downarrow}_{\mathbb{A}\mathrm{gt}}$. Moreover, we only use operators that refer to the currently available resources in the system.

structure, $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$ *is a set of agents;* $\pi : \Pi \to 2^Q$ *is a valuation of propositions;* Act *is a finite set of actions; and the function* $d : \mathbb{A}\mathrm{gt} \times Q \to 2^{Act} \setminus \{\emptyset\}$ *indicates the actions available to agent* $a \in \mathbb{A}\mathrm{gt}$ *at state* $q \in Q$*. We write* $d_a(q)$ *instead of* $d(a, q)$*, and use* $d(q)$ *to denote the set* $d_1(q) \times \ldots \times d_k(q)$ *of action profiles in state* q*. Similarly,* $d_A(q)$ *denotes the action tuples available to* A *at* q*.* o *is a transition function which maps each state* $q \in Q$ *and action profile* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k) \in d(q)$ *(specifying a move for each agent) to another state* $q' = o(q, \boldsymbol{\alpha})$*. Finally, the function* $t : Act \times \mathcal{R}es \to \mathbb{Z}$ *models the resources consumed and produced by actions. We define* $\mathsf{prod}(\alpha, r) := \max\{0, t(\alpha, r)\}$ *(resp.* $\mathsf{cons}(\alpha, r) := \min\{0, t(\alpha, r)\}$*) as the amount of resource* r *produced (resp. consumed) by action* α*. For* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$*, we use* $\boldsymbol{\alpha}_A$ *to denote the sub-tuple consisting of the actions of agents* $A \subseteq \mathbb{A}\mathrm{gt}$*.*

An **RBM** with idle actions*,* **iRBM** *for short, is an* **RBM** $\mathfrak{M}$ *such that for all agents* a*, all states* q*, there is an action* $\alpha \in d_a(q)$ *such that for all resource types* r *in* $\mathfrak{M}$ *we have that* $t(\alpha, r) = 0$*. We refer to this action (or to one of them if there is more than one) as the* idle action *of* a *and denote it by* $idle$*.*

A *path* $\lambda \in Q^\omega$ is an infinite sequence of states such that there is a transition between two adjacent states. A *resource-extended* path $\lambda \in (Q \times \mathsf{En})^\omega$ is an infinite sequence over $Q \times \mathsf{En}$ such that the restriction to states (the first component), denoted by $\lambda|_Q$, is a path in the underlying model. The projection of λ to the second component of each element in the sequence is denoted by $\lambda|_{\mathsf{En}}$. We define $\lambda[i]$ to be the $i+1$-th element of λ, and $\lambda[i, \infty]$ to be the suffix $\lambda[i]\lambda[i+1] \ldots$. A *strategy*[4] for a coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is a function $s_A : (Q \times \mathsf{En})^+ \to Act^A$ such that $s_A((q_0, \eta_0) \ldots (q_n, \eta_n)) \in d_A(q_n)$ for $(q_0, \eta_0) \ldots (q_n, \eta_n) \in (Q \times \mathsf{En})^+$. Such a strategy gives rise to a set of (resource-extended) paths that can emerge if agents follow their strategies. A (q, η, s_A)*-path* is a resource-extended path λ such that for all $i = 0, 1, \ldots$ with $\lambda[i] := (q_i, \eta_i)$ there is an action profile $\boldsymbol{\alpha} \in d(\lambda|_Q[i])$ such that:

1. $q_0 = q$ and $\eta_0(r) = \min\{\beta(r), \eta(r)\}$ for all $r \in \mathcal{R}es$ (describes initial configuration);
2. $s_A(\lambda[0, i]) = \boldsymbol{\alpha}_A$ (A follow their strategy);
3. $\lambda|_Q[i+1] = o(\lambda|_Q[i], \boldsymbol{\alpha})$ (transition according to $\boldsymbol{\alpha}$);
4. for all $\boldsymbol{\alpha}' \in Act_{\mathbb{A}\mathrm{gt} \setminus A}$ and for all $r \in \mathcal{R}es$: $\eta_i(r) \geq \sum_{a \in \mathbb{A}\mathrm{gt} \setminus A} \mathsf{cons}(\boldsymbol{\alpha}'_a, r) + \sum_{a \in A} \mathsf{cons}(\boldsymbol{\alpha}_a, r)$ (enough resources to perform the actions are available); and
5. $\eta_{i+1}(r) = \eta_i(r) + \sum_{a \in \mathsf{Agt}} \mathsf{prod}(\boldsymbol{\alpha}_a) - \sum_{a \in \mathsf{Agt}} \mathsf{cons}(\boldsymbol{\alpha}_a)$ for all $r \in \mathcal{R}es$.

Condition (iv) models that the opponents have priority when claiming resources.

The (q, η, s_A)*-outcome* of a strategy s_A in q, $out(q, \eta, s_A)$, is defined as the set of all (q, η, s_A)-paths starting in q. We also refer to this set as an *execution tree* of A. Truth is defined over an **RBM** $\mathfrak{M}$, a state $q \in Q$, and an endowment η. The *semantics* is given by the satisfaction relation $\models$ defined below.

$\mathfrak{M}, q, \eta \models \mathsf{p}$ iff $\mathsf{p} \in \Pi$ and $q \in \pi(\mathsf{p})$.
$\mathfrak{M}, q, \eta \models \varphi_1 \wedge \varphi_2$ iff $\mathfrak{M}, q, \eta \models \varphi_1$ and $\mathfrak{M}, q, \eta \models \varphi_2$

[4] We note that differently from [8,1,3], our notion of strategy takes the history of states as well as the history of endowments into account. In the setting considered here such strategies are more powerful than strategies only taking the state-component into account.

$\mathfrak{M}, q, \eta \models \neg\varphi$ iff it is not the case that $\mathfrak{M}, q, \eta \models \varphi$
$\mathfrak{M}, q, \eta \models \langle\!\langle A \rangle\!\rangle^{\downarrow} \mathbf{X}\varphi$ iff there is a strategy s_A for A such that for all $\lambda \in out(q, \eta, s_A)$, $\mathfrak{M}, \lambda_Q[1], \eta \models \varphi$
$\mathfrak{M}, q, \eta \models \langle\!\langle A \rangle\!\rangle^{\downarrow} \psi \mathbf{U} \varphi$ iff there exists a strategy s_A for A such that for all $\lambda \in out(q, \eta, s_A)$, there is an i with $i \geq 0$ and $\mathfrak{M}, \lambda|_Q[i], \lambda|_{\mathsf{En}}[i] \models \varphi$ such that for all j with $0 \leq j < i$ it holds that $\mathfrak{M}, \lambda|_Q[j], \lambda|_{\mathsf{En}}[j] \models \psi$
$\mathfrak{M}, q, \eta \models \langle\!\langle A \rangle\!\rangle^{\downarrow} \mathbf{G}\varphi$ iff there exists a strategy s_A for A such that for all $\lambda \in out(q, \eta, s_A)$ and all $i \geq 0$, $\mathfrak{M}, \lambda|_Q[i], \lambda|_{\mathsf{En}}[i] \models \varphi$

The *model checking problem* is to determine whether $\mathfrak{M}, q, \eta \models \varphi$ holds.

Example. We illustrate the framework by extending the introductory example on the departmental travelling budget. Consider a department which consists of a dean d, two professors p_1, p_2 and three lecturers $l_1, l_2,$ and l_3. The department's travel budget is allocated annually and can be spent to attend conferences. There are three categories to request money: premium, advanced, and economic. All options are available to the dean, the last two to professors, and only the last one to the lecturers. For instance, if the cost of attending PRIMA[5] is, depending on the category, \$2000, \$1000, and \$500, respectively, then with an available budget of \$4000 not all lecturers can be sure to be able to attend PRIMA. Because, the dean and the professors could all decide to attend PRIMA and to request the advanced category. In that case, only \$1000 would remain, not enough for all lecturers to attend; formally specified, $\langle\!\langle d, p_1, p_2 \rangle\!\rangle^{\downarrow} \mathbf{F}(\mathsf{d} \wedge \mathsf{p_1} \wedge \mathsf{p_2} \wedge \neg\langle\!\langle \{l_1, l_2, l_3\} \rangle\!\rangle^{\downarrow} \mathbf{F}(\mathsf{l_1} \wedge \mathsf{l_2} \wedge \mathsf{l_3}))$ is true where a proposition x expresses that "person" x is attending PRIMA. Equivalently, $\neg\langle\!\langle \{l_1, l_2, l_3\} \rangle\!\rangle^{\downarrow} \mathbf{F}(\mathsf{l_1} \wedge \mathsf{l_2} \wedge \mathsf{l_3}))$ is true; this highlights that the opponents have priority in claiming resources. However, by collaborating with the professors, they have a strategy which allows all lecturers to attend, independent of the actions of the dean: i.e., $\langle\!\langle \{p_1, p_2, l_1, l_2, l_3\} \rangle\!\rangle^{\downarrow} \mathbf{F}(\mathsf{l_1} \wedge \mathsf{l_2} \wedge \mathsf{l_3})$.

3 Model Checking CTL over Büchi Pushdown Systems

We first review existing results on alternating Büchi pushdown systems (ABPDSs). Then, we use these results to give an automata-theoretic approach to model check CTL-formulae over *compact* ABPDSs. The latter will be used to encode $\mathbf{RBM}$s in Section 4. An *alphabet* Γ is a non-empty, finite set of symbols. Γ^* denotes the set consisting of all finite words over Γ including the empty word ϵ. Typical symbols from Γ are denoted by $a, b, \ldots$ and words by $w, v, u, \ldots$. We read words from left to right. As before, we assume that Π denotes a finite, non-empty set of propositions.

3.1 Alternating Büchi Pushdown Systems

We use words to represent the stack content. We say that word $w = a_1 \ldots a_n$ is on the stack if a_1 is the lowest symbol, followed by a_2 and so forth. The symbol on top is a_n. An *alternating pushdown system* (APDS) is a tuple $\mathcal{P} = (P, \Gamma, \Delta)$ where P is a non-empty, finite set of control states, Γ a non-empty, finite (stack) alphabet,

[5] PRIMA is the acronym for the conference Principles and Pratice of Multi-Agent Systems. A short version of this paper was accepted for PRIMA 2015 [9].

and $\Delta \subseteq (P \times \Gamma) \times 2^{P \times \Gamma^*}$ a transition relation [5,16]. We call $\mathcal{P}$ a *pushdown system* (PDS) if $(s,a)\Delta X$ implies $|X| = 1$ where $X \in 2^{P \times \Gamma^*}$. An *alternating Büchi pushdown system* (ABPDS) $\mathcal{B} = (P, \Gamma, \Delta, F)$ is defined as a APDS but a set of accepting states $F \subseteq P$ is added. In the following we focus on ABPDSs, but most of the definitions do also apply to APDSs and PDSs with obvious changes. A transition $(p,a)\Delta\{(p_1,w_1),\ldots,(p_n,w_n)\}$ represents that if the system is in state p and the top-stack symbol is a then the ABPDS $\mathcal{B}$ is copied n-times where the ith copy changes its local state to p_i, pops a from the stack and pushes w_i on the stack, $1 \leq i \leq n$. For a transition rule $(p,a)\Delta\{(p_1,w_1),\ldots,(p_n,w_n)\}$ and a stack content $w \in \Gamma^*$ we say that (p,wa) is an *immediate predecessor* of $\{(p_1,ww_1),\ldots,(p_n,ww_n)\}$. We write $(p,wa) \Rightarrow_{\mathcal{B}} \{(p_1,ww_1),\ldots,(p_n,ww_n)\}$. We also say that $\{(p_1,ww_1),\ldots,(p_n,ww_n)\}$ is an *immediate successor* of (p,wa). We often write $(p,a)\Delta(p',w)$ for $(p,a)\Delta\{(p',w)\}$. Finally, we would like to note that a *stack bottom symbol* $\#$ can be defined the only purpose of which is to denote that the stack is empty. Apart from this the symbol is never touched. The introduction of $\#$ simply requires adding $\#$ to Γ and to add a rules which pushes $\#$ to the stack, before any other rule is applied. In the following we assume that $\#$ is the stack bottom symbol whenever it appears in the text.

A *configuration* of $\mathcal{B}$ is a tuple from $\mathsf{Cnf}_{\mathcal{B}} = P \times \Gamma^*$. A *c-run* ρ of $\mathcal{B}$, where c is a configuration of $\mathcal{B}$, is a tree in which each node is labelled by a configuration such that the root of the tree is labelled by c. If a node labelled by (p,w) has n (direct) child nodes labelled by $(p_1,w_1),\ldots,(p_n,w_n)$, respectively, then it is required that $(p,w) \Rightarrow_{\mathcal{B}} \{(p_1,w_1),\ldots,(p_n,w_n)\}$. We use $\mathcal{R}_{\mathcal{B}}(c)$ to denote the set of all c-runs[6] and $\mathcal{R}_{\mathcal{B}} = \bigcup_{c \in \mathsf{Cnf}_{\mathcal{B}}} \mathcal{R}_{\mathcal{B}}(c)$. We note that a run in a PDS $\mathcal{P}$ is simply a linear sequence of configurations. A *ρ-path*, $\rho \in \mathcal{R}_{\mathcal{B}}(c)$, is a maximal length branch $\kappa = c_0 c_1 \ldots$ of ρ starting at the root node c. We shall identify ρ with its set of paths and write $\kappa \in \rho$ to indicate that κ is a ρ-path. Again, in the case of a PDS $\mathcal{P}$ a run and a path in it are essentially the same. We say that $\kappa \in \rho$ is *accepting* if a state of F occurs infinitely often in configurations on κ. A run is accepting if each path $\kappa \in \rho$ is accepting; and a configuration c is accepting if there is an accepting run $\rho \in \mathcal{R}_{\mathcal{B}}(c)$. We note that an accepting run of an ABPDS has only infinite branches. The *language* accepted by $\mathcal{B}$, $L(\mathcal{B})$, is the set of all accepting configurations. Finally, we define $\Rightarrow^*_{\mathcal{B}} \subseteq (P \times \Gamma^*) \times 2^{P \times \Gamma^*}$ as, roughly speaking, the reflexive transitive closure of $\Rightarrow_{\mathcal{B}}$; that is, $c \Rightarrow^*_{\mathcal{B}} \{c\}$ for all c; if $c \Rightarrow_{\mathcal{B}} C$ then $c \Rightarrow^*_{\mathcal{B}} C$; and if $c \Rightarrow^*_{\mathcal{B}} \{c_1,\ldots,c_n\}$ and $c_i \Rightarrow^*_{\mathcal{B}} C_i$ for every $1 \leq i \leq n$, then $c \Rightarrow^*_{\mathcal{B}} \bigcup_i C_i$.

A nice property of an ABPDS is that its set of accepting configurations is regular, in the sense that it is accepted by an appropriate automaton which is defined next. An *alternating automaton* [5] is a tuple $\mathcal{A} = (S, \Sigma, \delta, I, S_f)$ where S is a finite, non-empty set of states, $\delta \subseteq S \times \Sigma \times 2^S$ is a transition relation, Σ an input alphabet, $I \subseteq S$ a set of initial states, and $S_f \subseteq S$ a set of final states. Similar to $\Rightarrow^*_{\mathcal{B}}$ we define the reflexive, transitive transition relation $\rightarrow^*_{\mathcal{A}} \subseteq (S \times \Sigma^*) \times 2^S$ as follows (where we write $s \xrightarrow{w}{}^*_{\mathcal{A}} S'$ for $(s,w,S') \in \rightarrow^*_{\mathcal{A}}$): $s \xrightarrow{\epsilon}{}^*_{\mathcal{A}} \{s\}$, if $(s,a,S') \in \delta$ then $s \xrightarrow{a}{}^*_{\mathcal{A}} S'$, and if $s \xrightarrow{w}{}^*_{\mathcal{A}} \{s_1,\ldots,s_n\}$ and $s_i \xrightarrow{a}{}^*_{\mathcal{A}} S_i$ for $1 \leq i \leq n$ then $s \xrightarrow{wa}{}^*_{\mathcal{A}} \bigcup_i S_i$. The automaton

[6] Sometimes, we assume that elements in X in a transition $(p,a)\Delta X$ are ordered and correspondently the branches in a run.

accepts $(s, w) \in S \times \Sigma^*$ iff $s \xrightarrow{w}{}^*_{\mathcal{A}} S'$ with $S' \subseteq S_f$ and $s \in I$. The language accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. A language is called *regular* if it is accepted by an alternating automaton. Finally, for a given ABPDS $\mathcal{B} = (P, \Gamma, \Delta, F)$ we define an *alternating $\mathcal{B}$-automaton* as an alternating automaton $(S, \Sigma, \delta, I, S_f)$ such that $I \subseteq P \subseteq S$ and $\Gamma = \Sigma$. We recall the following result from [15]:

Theorem 1 ([15]). *For any ABPDS $\mathcal{B}$ there is an effectively computable alternating $\mathcal{B}$-automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{B})$.*

The authors of [15] do also determine the size of the automaton. As we are not concerned with the computational complexity in this paper, we omit these results.

3.2 Model Checking CTL over PDSs

The Logic CTL. Computation Tree Temporal Logic (CTL) [12] can be seen as the one agent, non-resource-constrained variant of RAL. Formulae of the logic are defined by the grammar: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\mathbf{X}\varphi \mid \mathsf{E}\mathbf{G}\varphi \mid \mathsf{E}(\varphi\mathbf{U}\psi)$ where $\mathsf{p} \in \Pi$. E denotes the existential path quantifier. $\mathsf{E}\varphi$ expresses that there is a run on which φ holds. The Boolean connectives are given by their usual abbreviations. In addition to that, we define the macros $\mathbf{F}\varphi \equiv \top\mathbf{U}\varphi$, $\mathsf{A}\mathbf{X}\varphi \equiv \neg\mathsf{E}\mathbf{X}\neg\varphi$, $\mathsf{A}\mathbf{G}\varphi \equiv \neg\mathsf{E}\mathbf{F}\neg\varphi$, and $\mathsf{A}\varphi\mathbf{U}\psi \equiv \neg\mathsf{E}((\neg\psi)\mathbf{U}(\neg\varphi \wedge \neg\psi)) \wedge \neg\mathbf{G}\neg\psi$. Thus, $\mathsf{A}\varphi$ is read as φ holds on all runs. The other temporal operators have the same meaning as for RAL. Moreover, for our constructions it is assumed that CTL-formulae are in *negation normal form*, that is negation only occurs at the propositional level. This makes it necessary to allow the connective $\vee$ (*or*) and also the *release operator* $\mathbf{R}$ as first-class citizens in the object language. Therefore, we use the following macros: $\mathsf{A}\varphi_1\mathbf{R}\varphi_2 = \neg\mathsf{E}(\neg\varphi_1)\mathbf{U}(\neg\varphi_2)$ and $\mathsf{E}\varphi_1\mathbf{R}\varphi_2 = \neg\mathsf{A}(\neg\varphi_1)\mathbf{U}(\neg\varphi_2)$. A subformula of a formula φ is a formula that occurs in φ, including φ itself. The *closure of* φ, $\mathsf{cl}(\varphi)$, is the set of all subformulae of φ. We define the set $\Pi^+(\varphi) = \{\mathsf{p} \in \Pi \mid \mathsf{p} \in \mathsf{cl}(\varphi)\}$ and $\Pi^-(\varphi) = \{\mathsf{p} \in \Pi \mid \neg\mathsf{p} \in \mathsf{cl}(\varphi)\}$ containing all propositional variables that occur positively and negatively in φ, respectively. Later, we also need a special closure $\mathsf{cl}_{\mathbf{R}}(\varphi)$ which consists of all formulae of the form $\mathsf{A}(\varphi_1\mathbf{R}\varphi_2)$ or $\mathsf{E}(\varphi_1\mathbf{R}\varphi_2)$ of $\mathsf{cl}(\varphi)$.

Model Checking over Pushdown Systems. The problem of CTL model checking over PDSs has been considered in, e.g., [15,5,6]. We now recall from [15] how the problem is defined. First, the PDS is extended with a labelling function lab to give truth to propositional atoms. In [15] two alternatives are considered. The first alternative assigns states to propositions, $\mathsf{lab} : \Pi \to 2^P$. The second alternative assigns configurations to propositions, $\mathsf{lab} : \Pi \to 2^{P \times \Gamma^*}$. In the following we only consider the second, more general alternative as this is the one we shall need for model checking RAL. For this type of labelling function we need a finite representation. We call lab *regular* if there is an alternating automaton $\mathcal{A}_\mathsf{p}$ with $L(\mathcal{A}_\mathsf{p}) = \mathsf{lab}(\mathsf{p})$ for each $\mathsf{p} \in \Pi$. We are ready to give the semantics of CTL-formulae over a PDS $\mathcal{P} = (P, \Gamma, \Delta)$, $c \in \mathsf{Cnf}_\mathcal{P}$, and a regular labelling function $\mathsf{lab} : \Pi \to 2^{P \times \Gamma^*}$. The semantics is defined by $\models$ as follows:

$\mathcal{P}, c, \mathsf{lab} \models \mathsf{p}$ iff $c \in \mathsf{lab}(\mathsf{p})$:
$\mathcal{P}, c, \mathsf{lab} \models \neg\varphi$ iff it is not the case that $\mathcal{P}, c, \mathsf{lab} \models \varphi$:
$\mathcal{P}, c, \mathsf{lab} \models \varphi_1 \wedge \varphi_2$ iff it is the case that $\mathcal{P}, c, \mathsf{lab} \models \varphi_1$ and $\mathcal{P}, c, \mathsf{lab} \models \varphi_2$:

$\mathcal{P}, c, \mathsf{lab} \models \mathsf{EX}\varphi$ iff there is a c-run $\rho = c_0 c_1, \ldots \in \mathcal{R}_{\mathcal{P}}(c)$ such that $\mathcal{P}, c_1, \mathsf{lab} \models \varphi$:
$\mathcal{P}, c, \mathsf{lab} \models \mathsf{EG}\varphi$ iff there is a c-run $\rho = c_0 c_1, \ldots \in \mathcal{R}_{\mathcal{P}}(c)$ such that $\mathcal{P}, c_i, \mathsf{lab} \models \varphi$ for all $i \geq 0$:
$\mathcal{P}, c, \mathsf{lab} \models \mathsf{E}\varphi_1 \mathbf{U} \varphi_2$ iff there is a c-run $\rho = c_0 c_1, \ldots \in \mathcal{R}_{\mathcal{P}}(c)$ such that there is an $i \in \mathbb{N}_0$ with $\mathcal{P}, c_i, \mathsf{lab} \models \varphi_2$ and for all $0 \leq j < i$ we have that $\mathcal{P}, c_j, \mathsf{lab} \models \varphi_1$.

The authors of [15] give a model checking algorithm which uses ABPDSs. They construct from $\mathcal{P}$, lab and φ, an ABPDS $\mathcal{B}_{\mathcal{P},\varphi}$ such that $\mathcal{P}, (p, w), \mathsf{lab} \models \varphi$ iff $((p, \varphi), w) \in L(\mathcal{B}_{\mathcal{P},\varphi})$. The ABPDS is essentially the product of the PDS $\mathcal{P}$ with the closure of φ, in particular states of $\mathcal{B}_{\mathcal{P},\varphi}$ are tuples $(p, \psi) \in P \times \mathsf{cl}(\varphi)$. The existential and universal path quantifiers of the formula cause the alternation of the ABPDS. We will give more details in Section 3.3 where we consider model checking CTL-formulae over ABPDSs. We finish this section by recalling the following theorem which follows from Theorem 1.

Theorem 2 ([15]). *For a given PDS $\mathcal{P}$, a regular labelling function* lab*, and a* CTL*-formula φ there is an effectively computable alternating automaton $\mathcal{A}_{\mathcal{P},\varphi}$ such that for all configurations $c = (p, w) \in \mathsf{Cnf}_{\mathcal{P}}$ the following holds: $\mathcal{P}, c, \mathsf{lab} \models \varphi$ iff $((p, \varphi), w) \in L(\mathcal{A}_{\mathcal{P},\varphi})$.*

3.3 Model Checking CTL over ABPDS

For our later results, we need to be able to define the truth of CTL-formulae over ABPDSs rather than PDSs. We extend the result of Theorem 2 accordingly. Let an ABPDS $\mathcal{B}$ be given. We first discuss what it means that $\mathcal{B}, c, \mathsf{lab} \models \mathsf{E}\varphi$. As before, we interpret it as: there is a *run* $\rho \in \mathcal{R}_{\mathcal{B}}(c)$ on which φ holds. However, given that ρ is a tree in the case of ABPDSs (or a set of paths) we need to explain how to evaluate φ on trees. We require that φ must be true on *each* path $\kappa \in \rho$ on the run. This can nicely be illustrated if $\mathcal{B}$ is considered as a two player game where player one decides which transition to take, and player two selects one of the child states. Thus, $\mathsf{E}\varphi$ expresses that player one has a strategy in the sense that it can enforce a run ρ such that player two cannot make φ false on any path $\kappa \in \rho$. The precise semantics is given next where $c \in \mathsf{Cnf}_{\mathcal{B}}$, and lab is a regular labelling function as before:

$\mathcal{B}, c, \mathsf{lab} \models \mathsf{p}$ iff $c \in \mathsf{lab}(\mathsf{p})$;
$\mathcal{B}, c, \mathsf{lab} \models \neg\varphi$ iff it is not the case that $\mathcal{B}, c, \mathsf{lab} \models \varphi$;
$\mathcal{B}, c, \mathsf{lab} \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{B}, c, \mathsf{lab} \models \varphi_1$ and $\mathcal{B}, c, \mathsf{lab} \models \varphi_2$;
$\mathcal{B}, c, \mathsf{lab} \models \mathsf{EX}\varphi$ iff there is a c-run $\rho \in \mathcal{R}_{\mathcal{B}}(c)$ such that for all paths $c_0 c_1 \ldots \in \rho$ it holds that $\mathcal{B}, c_1, \mathsf{lab} \models \varphi$;
$\mathcal{B}, c, \mathsf{lab} \models \mathsf{EG}\varphi$ iff there is a c-run $\rho \in \mathcal{R}_{\mathcal{B}}(c)$ such that for all paths $c_0 c_1 \ldots \in \rho$ it holds that $\mathcal{B}, c_i, \mathsf{lab} \models \varphi$ for all $i \geq 0$;
$\mathcal{B}, c, \mathsf{lab} \models \mathsf{E}\varphi_1 \mathbf{U} \varphi_2$ iff there is a c-run $\rho \in \mathcal{R}_{\mathcal{B}}(c)$ such that for all paths $c_0 c_1 \ldots \in \rho$ there is an $i \in \mathbb{N}_0$ with $\mathcal{B}, c_i, \mathsf{lab} \models \varphi_2$ and for all $0 \leq j < i$ we have that $\mathcal{B}, c_j, \mathsf{lab} \models \varphi_1$.

We are ready to give a variant of Theorem 2 over ABPDSs. It follow from Lemma 1 given below in combination with Theorem 1.

Theorem 3. *For a given ABPDS $\mathcal{B}$, a regular labelling function* lab*, and a* CTL*-formula φ there is an effectively computable alternating automaton $\mathcal{A}_{\mathcal{B},\varphi}$ such that for all configurations $c = (p, w) \in \mathsf{Conf}_{\mathcal{B}}$ the following holds: $\mathcal{B}, c, \mathsf{lab} \models \varphi$ iff $((p, \varphi), w) \in L(\mathcal{A}_{\mathcal{B},\varphi})$.*

The proof of the theorem is closely related to the proof of Theorem 2 given in [15]. Here, however, the branching in the resulting ABPDSs can have two different sources: branching can result from branching in the input ABPDS or from the universal CTL-path quantifier. We sketch the construction of $\mathcal{B}_\varphi$ for a given ABPDS $\mathcal{B} = (P, \Gamma, \Delta, F)$, a regular labelling function $\mathsf{lab} : \Pi \to 2^{P \times \Gamma^*}$, and a CTL-formula φ.

First, for each $\mathsf{p} \in \Pi$ let $\mathcal{A}_\mathsf{p} = (S_\mathsf{p}, \Gamma, \delta_\mathsf{p}, I_\mathsf{p}, F_\mathsf{p})$ denote the alternating $\mathcal{B}$-automaton that accepts $L(\mathcal{A}_\mathsf{p}) = \mathsf{lab}(\mathsf{p})$, and $\mathcal{A}_{\neg\mathsf{p}}$ be the alternating automaton with $L(\mathcal{A}_{\neg\mathsf{p}}) = P \times \Gamma^* \backslash \mathsf{lab}(\mathsf{p})$. Due to technical reasons, we make the state spaces disjoint. We add p as subindex to every state of S_p; for example, a state p becomes p_p. Note, that it then holds that $L(\mathcal{A}_\mathsf{p}) \subseteq P_\mathsf{p} \times \Gamma^*$ rather than $L(\mathcal{A}_\mathsf{p}) \subseteq P \times \Gamma^*$ where P_p is the set of renamed states of P. In particular, the automaton $\mathcal{B}_\varphi$ will include states (p, p) which are connected to an initial state of the form $\mathcal{A}_\mathsf{p}$. This initial state of $\mathcal{A}_\mathsf{p}$ is denoted by p_p; we proceed similarly for $\mathcal{A}_{\neg\mathsf{p}}$. The ABPDS $\mathcal{B}_\varphi = (P', \Gamma, \Delta', F')$ is defined as follows (cf. Section 3.2 for the notation used):

- $P' = (P \times \mathsf{cl}(\varphi)) \cup \bigcup_{\mathsf{p} \in \Pi^+(\varphi)} S_\mathsf{p} \cup \bigcup_{\mathsf{p} \in \Pi^-(\varphi)} S_{\neg\mathsf{p}}$
- $F' = (P \times \mathsf{cl}_\mathbf{R}(\varphi)) \cup \bigcup_{\mathsf{p} \in \Pi^+(\varphi)} F_\mathsf{p} \cup \bigcup_{\mathsf{p} \in \Pi^-(\varphi)} F_{\neg\mathsf{p}}$
- $\Delta' \subseteq (P' \times \Gamma) \times 2^{P' \times \Gamma^*}$ is the smallest relation such that:
  1. $((p, \mathsf{p}), a)\Delta'(p_\mathsf{p}, a)$ for $\mathsf{p} \in \Pi$
  2. $((p, \neg\mathsf{p}), a)\Delta'(p_{\neg\mathsf{p}}, a)$ for $\mathsf{p} \in \Pi$
  3. $((p, \varphi \wedge \psi), a)\Delta'\{(p, \varphi), a), ((p, \psi), a)\}$
  4. $((p, \varphi \vee \psi), a)\Delta'\{(p, \xi), a)\}$ for $\xi \in \{\varphi, \psi\}$
  5. $((p, \mathsf{E}\mathbf{X}\varphi), a)\Delta'\{((p', \varphi), w') \mid (p', w') \in X\}$ for each $(p, a)\Delta X$
  6. $((p, \mathsf{A}\mathbf{X}\varphi), a)\Delta' \bigcup_{(p,a)\Delta X}\{((p', \varphi), w') \mid (p', w') \in X\}$
  7. $((p, \mathsf{E}\varphi\mathbf{U}\psi), a)\Delta'((p, \psi), a)$
  8. $((p, \mathsf{E}\varphi\mathbf{U}\psi), a)\Delta'\{((p, \varphi), a)\} \cup \{((p', \mathsf{E}\varphi\mathbf{U}\psi), w') \mid (p', w') \in X\}$ for each $(p, a)\Delta X$
  9. $((p, \mathsf{A}\varphi\mathbf{U}\psi), a)\Delta'((p, \psi), a)$
  10. $((p, \mathsf{A}\varphi\mathbf{U}\psi), a)\Delta'\{((p, \varphi), a)\} \cup \bigcup_{(p,a)\Delta X}\{((p', \mathsf{A}\varphi\mathbf{U}\psi), w') \mid (p', w') \in X\}$
  11. $((p, \mathsf{E}\varphi\mathbf{R}\psi), a)\Delta'((p, \varphi), a)$
  12. $((p, \mathsf{E}\varphi\mathbf{R}\psi), a)\Delta'\{((p, \psi), a)\} \cup \{((p', \mathsf{E}\varphi\mathbf{R}\psi), w') \mid (p', w') \in X\}$ for each $(p, a)\Delta X$
  13. $((p, \mathsf{A}\varphi\mathbf{R}\psi), a)\Delta'((p, \varphi), a)$
  14. $((p, \mathsf{A}\varphi\mathbf{R}\psi), a)\Delta'\{((p, \psi), a)\} \cup \bigcup_{(p,a)\Delta X}\{((p', \mathsf{A}\varphi\mathbf{R}\psi), w') \mid (p', w') \in X\}$
  15. If $(s, a, S') \in (\bigcup_{\mathsf{p} \in \Pi^+(\varphi)} \delta_\mathsf{p} \cup \bigcup_{\mathsf{p} \in \Pi^-(\varphi)} \delta_{\neg\mathsf{p}})$ then $(s, a)\Delta'\{(s', \epsilon) \mid s' \in S'\}$
  16. If $s \in (\bigcup_{\mathsf{p} \in \Pi^+(\varphi)} F_\mathsf{p} \cup \bigcup_{\mathsf{p} \in \Pi^-(\varphi)} F_{\neg\mathsf{p}})$ then $(s, \#)\Delta'(s, \#)$

Intuitively, $\mathcal{B}_\varphi$ simulates the semantics of CTL over ABPDSs and keeps track of the formulae to be satisfied. Let us consider rule 5 and suppose that the current configuration of $\mathcal{B}_\varphi$ is $((p, \mathsf{E}\mathbf{X}\psi), wa)$. Then, $\mathcal{B}_\varphi$ selects one set X (this models the existential

quantifier E) with $(p,a)\Delta X$ and sends a copy of $\mathcal{B}_\varphi$ to each of the successor state in X (this models the temporal operator **X**). Rule 15 is responsible for simulating the alternating automaton at the propositional level and 16 ensures that the acceptance state of the alternating automaton also yields an accepting path of $\mathcal{B}_\varphi$. The final states include states of type $P \times \mathsf{cl}_{\mathbf{R}}(\varphi)$ to ensure that formulae are accepted which are *never released*. For further details of the standard functioning we refer to [15]. We note that rules 5-14 had to be modified to work with ABPDSs. For a proof sketch of the next Lemma, we refer to the more general Lemma 2.

Lemma 1. *Using the notation above, we have the following:* $\mathcal{B}, (p,w), \mathsf{lab} \models \psi$ *iff* $((p,\psi),w) \in L(\mathcal{B}_\varphi)$ *for all* $\psi \in \mathsf{cl}(\varphi)$.

3.4 Compact ABPDS

Our reduction of model checking RAL to an acceptance problem over ABPDSs relies on an encoding of an 1-unbounded $\mathbf{iRBM}$ as an ABPDS. Roughly speaking, the stack is used to keep track of the shared pool of resources. A technical difficulty is that an action may consume several resources at a time, whereas an ABPDS can only read the top stack symbol. Therefore, we introduce a more compact encoding of an ABPDS which allows to read (and pop) more than one stack symbol at a time.

Given a natural number $r \geq 1$, an *r-compact* ABPDS (CABPDS) is a tuple $\mathcal{C} = (P, \Gamma, \Delta, F, r)$ where all ingredients have the same meaning as in an ABPDS with the exception that $\Delta \subseteq P \times \Gamma^{\leq r} \times 2^{P \times \Gamma^*}$ where $\Gamma^{\leq r} = \bigcup_{i=1}^{r} \Gamma^i$ denotes the set of all non-empty words over Γ of length at most r. This models that the selection of the next transition can depend on up to the top r stack symbols. All notions introduced so far are also used for CABPDSs. Note that in a configuration $(p, a_1 \dots a_n)$ a transition $(p, b_1 \dots b_j)\Delta\{(p_1,w_1),\dots,(p_m,w_m)\}$ can only be taken if, and only if, $n \geq j$ and $a_{n-j+1} \dots a_n = b_1 \dots b_j$. In that case $(p, a_1 \dots a_n) \Rightarrow_{\mathcal{C}} \{(p_1, a_1 \dots a_{n-j} w_1), \dots, (p_m, a_1 \dots a_{n-j} w_m)\}$. Obviously, a 1-compact ABPDS is simply an ABPDS.

A CABPDS is no more expressive than a "standard" ABPDS. Essentially, the top r stack symbols can be encoded in the states of an ABPDS. We make this intuition precise. Let $\mathcal{C}$ be the r-compact ABPDS given above. We define the ABPDS $\mathcal{B}(\mathcal{C}) = (P', \Gamma, \Delta', F)$ consisting of the following elements:

- $P' = P \cup P \times \Gamma^{\leq r} \times \Gamma^{\leq r-1}$ where states in P are called *real states*, and all other states *storage states*. A storage state has the form (p,w,v) and encodes that word w should be popped from the stack where its prefix v remains to be popped.
- $\Delta' \subseteq P' \times \Gamma \times 2^{P' \times \Gamma^*}$ is the smallest relation satisfying the following properties: For all $p \in P, a \in \Gamma, w, v \in \Gamma^+$ and $X \subseteq P \times \Gamma^*$:
  1. If $(p,a)\Delta X$ then $(p,a)\Delta' X$;
  2. If $(p,wa)\Delta X$ then $(p,a)\Delta'((p,wa,w),\epsilon)$;
  3. $((p,vaw,va),a)\Delta'((p,vaw,v),\epsilon)$ provided that $(p,vaw)\Delta X$;
  4. $((p,aw,a),a)\Delta' X$ provided that $(p,aw)\Delta X$.

We briefly explain these conditions. When a transition of $\mathcal{C}$ pops only a single symbol from the stack, it is also a transition in $\mathcal{B}(\mathcal{C})$ (rule 1). If a transition of $\mathcal{C}$ pops more than one symbol the transition is split into several in $\mathcal{B}(\mathcal{C})$. The first symbol and the

transition to a storage state is described by Condition 2: the top symbol a is popped from the stack and the next state is (p, wa, w). It expresses that wa should be popped and that w remains to be popped (a has been popped by this very rule); note that ϵ in this transition means no symbol is pushed on the stack. Condition 3 describes how to pop a single symbol a from a storage node where va is the word which remains to be popped in order to complete the simulation of the transition $(p, vaw)\Delta X$. Finally, the last rule is applied if all but the last symbols a of the transition $(p, aw)\Delta X$ that is being simulated is read and can be popped from the stack. This completes the encoding and we obtain the following result:

Proposition 1. *For any r-compact ABPDS $\mathcal{C} = (P, \Gamma, \Delta, F, r)$ we have that $c \in L(\mathcal{C})$ iff $c \in L(\mathcal{B}(\mathcal{C}))$, for all configurations $c \in P \times \Gamma^*$.*

Proof. In the following, we define a function f which translates a run $\rho \in \mathcal{R}_{\mathcal{C}}$ into one of $\mathcal{R}_{\mathcal{B}(\mathcal{C})}$. Let $\rho = (p, w)(\rho_1, \ldots, \rho_n)$[7] and (p_i, w_i) be the root of ρ_i. This means $(p, w) \Rightarrow_{\mathcal{C}} \{(p_i, w_i) \mid 1 \leq i \leq n\}$. Then, f is defined by induction on the structure of ρ as follows:

- if $(p, w) \Rightarrow_{\mathcal{C}} \{(p_i, w_i) \mid 1 \leq i \leq n\}$ is generated by $(p, a)\Delta\{(p_i, w_i') \mid 1 \leq i \leq n\}$ for some $a \in \Gamma$, then $f(\rho) = (p, w)(f(\rho_1), \ldots, f(\rho_n))$; and
- if $(p, w) \Rightarrow_{\mathcal{C}} \{(p_i, w_i) \mid 1 \leq i \leq n\}$ is generated by $(p, a_1 \ldots a_m)\Delta\{(p_i, w_i') \mid 1 \leq i \leq n\}$ for some $m > 1$ and $a_1, \ldots, a_m \in \Gamma$ (i.e., $w = w'a_1 \ldots a_m$), then $f(\rho) = (p, w)(((p, a_1 \ldots a_m, a_1 \ldots, a_{m-1}), w'a_1 \ldots a_{m-1})(\ldots(((p, a_1 \ldots a_m, a_1), w'a_1)(f(\rho_1), \ldots, f(\rho_n)))\ldots))$.

Furthermore, for any path $\kappa \in f(\rho)$, we can pinpoint, by abuse of notation, the corresponding path $f^{-1}(\kappa)$ in ρ as

- $f^{-1}((p, w)(p_i, w_i)\ldots) = (p, w)f^{-1}((p_i, w_i)\ldots)$; and
- $f^{-1}((p, w)((p, a_1 \ldots a_m, a_1 \ldots a_{m-1}), w'a_1 \ldots a_{m-1})\ldots((p, a_1 \ldots a_m, a_1), w'a_1)(p_i, w_i)\ldots) = (p, w)f^{-1}((p_i, w_i)\ldots)$.

Obviously, any state occurring infinitely often in $f^{-1}(\kappa)$ also appears infinitely often in κ.

($\Rightarrow$) : Assume that $c \in L(\mathcal{C})$, then there exists an accepting run $\rho \in \mathcal{R}_{\mathcal{C}}(c)$. Then $f(\rho) \in \mathcal{R}_{\mathcal{B}(\mathcal{C})}(c)$. For each $\kappa \in f(\rho)$, $f^{-1}(\kappa)$ is accepting, i.e., some state in F occurs infinitely often in κ; hence, it also occurs infinitely often in κ, showing that $f(\rho)$ is accepting, i.e., $c \in L(\mathcal{B}(\mathcal{C}))$.

($\Leftarrow$) : The proof is done analogously to the above case where the function f^{-1} is used to translate an accepting run $\rho \in \mathcal{R}_{\mathcal{B}(\mathcal{C})}(c)$ into that of $\mathcal{R}_{\mathcal{C}}(c)$. □

The next corollary is easy to see: the language $L(\mathcal{C}) = L(\mathcal{B}(\mathcal{C}))$ is regular as $L(\mathcal{B}(\mathcal{C}))$ and $P \times \Gamma^*$ are regular and regular languages are closed under intersection.

Corollary 1. *$L(\mathcal{C})$ is regular.*

[7] We denote with $c(\rho_1, \ldots, \rho_n)$ a tree with root c which has n direct sub-tree $\rho_1, \ldots, \rho_n$ starting at the child nodes of c.

3.5 CTL Model checking over compact ABPDSs

In this section, we consider model checking CTL over CABPDSs. Given an r-compact ABPDS $\mathcal{C} = (P, \Gamma, \Delta, F, r)$, a regular labelling function $\mathsf{lab} : \Pi \to 2^{P \times \Gamma^*}$ and a CTL-formula φ over Π. Assume that for each $a \in \Pi^+(\varphi)$, $\mathsf{lab}(a)$ is accepted by an alternating automaton $\mathcal{A}_\mathsf{p} = (S_\mathsf{p}, \Gamma, \delta_\mathsf{p}, I_\mathsf{p}, F_\mathsf{p})$; and for each $\mathsf{p} \in \Pi^-(\varphi)$, the complement $P \times \Gamma^* \setminus \mathsf{lab}(\mathsf{p})$ is accepted by an alternating automaton $\mathcal{A}_{\neg\mathsf{p}} = (S_{\neg\mathsf{p}}, \Gamma, \delta_{\neg\mathsf{p}}, I_{\neg\mathsf{p}}, F_{\neg\mathsf{p}})$. We assume the same notational conventions wrt. disjointness and renaming as discussed in Section 3.3. We define the r-compact ABPDS $\mathcal{C}_\varphi = (P', \Gamma, \Delta', F', r)$ as follows:

- $P' = (P \times \mathsf{cl}(\varphi)) \cup \bigcup_{\mathsf{p} \in \Pi^+(\varphi)} S_\mathsf{p} \cup \bigcup_{\mathsf{p} \in \Pi^-(\varphi)} S_{\neg\mathsf{p}}$;
- $F' = (P \times \mathsf{cl}_\mathbf{R}(\varphi)) \cup \bigcup_{\mathsf{p} \in \Pi^+(\varphi)} F_\mathsf{p} \cup \bigcup_{\mathsf{p} \in \Pi^-(\varphi)} F_{\neg\mathsf{p}}$; and
- Δ' is the smallest relation satisfying rules 1-14, 16 given in Section 3.3 where symbol a is replaced by word w everywhere, and rule 15 of Section 3.3 is taken without any changes.

The intuition of the construction of $\mathcal{C}_\varphi$ is the same as for $\mathcal{B}_\varphi$. We obtain the result:

Lemma 2. *Using the notation above, we have the following:* $\mathcal{C}, (p, w), \mathsf{lab} \models \psi$ *iff* $((p, \psi), w) \in L(\mathcal{C}_\varphi)$ *for all* $\psi \in \mathsf{cl}(\varphi)$.

Proof (Sketch). ($\Rightarrow$) : Assume that $\mathcal{C}, (p, w), \mathsf{lab} \models \psi$. We prove that $((p, \psi), w)$ has an accepting run in $\mathcal{C}_\varphi$ by induction on the structure of ψ.

Case $\psi = \mathsf{p}$**:** Since $\mathcal{C}, (p, w), \mathsf{lab} \models \psi$, $(p, w) \in \mathsf{lab}(\mathsf{p})$. Then, in $\mathcal{A}_\mathsf{p}$ we have $p_\mathsf{p} \xrightarrow{w}{}^*_{\mathcal{A}_\mathsf{p}} S'$ where $p_\mathsf{p} \in I_\mathsf{p}$ and $S' \subseteq F_\mathsf{p} \subseteq F'$. Then, $((p, \mathsf{p}), w) \Rightarrow_{\mathcal{C}_\varphi} (p_\mathsf{p}, w) \Rightarrow^*_{\mathcal{C}_\varphi} \{(s', \#) \mid s' \in S'\} \Rightarrow_{\mathcal{C}_\varphi} \{(s', \#) \mid s' \in S'\} \Rightarrow_{\mathcal{C}_\varphi} \ldots$ (by rules 1, 15 and 16) which is an accepting run in $\mathcal{R}_{\mathcal{C}_\varphi}$.

Case $\psi = \psi_1 \vee \psi_2$**:** Since $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1 \vee \psi_2$, $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1$ or $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_2$. Without loss of generality, let us assume that $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1$. By induction hypothesis, we have $((p, \psi_1), w) \in L(\mathcal{C}_\varphi)$, i.e., there is an accepting $((p, \psi_1), w)$-run ρ. Then, we construct a $((p, \psi_1 \vee \psi_2), w)$-run as $((p, \psi_1 \vee \psi_2), w)(\rho)$ which is obviously accepting. Hence, $((p, \psi_1 \vee \psi_2), w) \in L(\mathcal{C}_\varphi)$.

Case $\psi = \psi_1 \wedge \psi_2$**:** Since $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1 \wedge \psi_2$, $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1$ and $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_2$. By induction hypothesis, we have $((p, \psi_1), w) \in L(\mathcal{C}_\varphi)$ and $((p, \psi_2), w) \in L(\mathcal{C}_\varphi)$, i.e., there exist a $((p, \psi_1), w)$-run ρ_1 and a $((p, \psi_2), w)$-run ρ_2 which are both accepting. Then, we construct a $((p, \psi_1 \wedge \psi_2), w)$-run as $((p, \psi_1 \wedge \psi_2), w)(\rho_1, \rho_2)$ which is obviously accepting. Hence, $((p, \psi_1 \wedge \psi_2), w) \in L(\mathcal{C}_\varphi)$.

Case $\psi = \mathsf{EX}\psi_1$**:** Since $\mathcal{C}, (p, w), \mathsf{lab} \models \psi$, there exists a (p, w)-run $\rho = (p, w)(\rho_1, \ldots, \rho_n)$[8] for all roots (p_i, w_i) of ρ_i, $\mathcal{C}, (p_i, w_i), \mathsf{lab} \models \psi_1$. By induction hypothesis, for all $1 \leq i \leq n$, we have $((p_i, \psi_1), w_i) \in L(\mathcal{C}_\varphi)$, i.e., there exists an accepting $((p_i, \psi_1), w_i)$-run ρ'_i. Then, we construct a $((p, \mathsf{EX}\psi_1), w)$-run as $((p, \mathsf{EX}\psi_1), w)(\rho'_1, \ldots, \rho'_n)$ which is obviously accepting. Hence, $((p, \mathsf{EX}\psi_1), w) \in L(\mathcal{C}_\varphi)$.

Case $\psi = \mathsf{E}\psi_1\mathbf{U}\psi_2$**:** Since $\mathcal{C}, (p, w), \mathsf{lab} \models \psi$, there exists a (p, w)-run ρ such that, for all paths $\kappa = (p_0^\kappa, w_0^\kappa)(p_1^\kappa, w_1^\kappa) \ldots \in \rho \in \mathcal{R}_\mathcal{C}((p, w))$, $\exists i_\kappa \geq 0$ such that $\mathcal{C}, (p_{i_\kappa}^\kappa, w_{i_\kappa}^\kappa)$,

[8] Recall that $c(\rho_1, \ldots, \rho_n)$ denotes a tree which is rooted at c and has n direct sub-trees $\rho_1, \ldots, \rho_n$.

$\mathsf{lab} \models \psi_2$ and $\forall 0 \leq j < i_\kappa$ we have that $\mathcal{C}, (p_j^\kappa, w_j^\kappa), \mathsf{lab} \models \psi_1$. By induction hypothesis, we have $((p_{i_\kappa}^\kappa, \psi_2), w_{i_\kappa}^\kappa) \in L(\mathcal{C}_\varphi)$ and $((p_j^\kappa, \psi_1), w_j^\kappa) \in L(\mathcal{C}_\varphi)$ for all $0 \leq j < i_\kappa$. Now, we show that $((p_i^\kappa, \psi), w_i^\kappa) \in L(\mathcal{C}_\varphi)$ for all $\kappa \in \rho$ and $0 \leq i \leq i_\kappa$ by induction on $i_\kappa - i$ (the claim follows for $i = 0$, then we have $((p, \psi), w) \in L(\mathcal{C}_\varphi)$):

Base case: Assume that $i = i_\kappa$, then $((p_{i_\kappa}^\kappa), \psi), w_{i_\kappa}^\kappa) \Rightarrow_{\mathcal{C}_\varphi} ((p_{i_\kappa}^\kappa, \psi_2), w_{i_\kappa}^\kappa)$ by rule 7. Since $((p_{i_\kappa}^\kappa, \psi_2), w_{i_\kappa}^\kappa) \in L(\mathcal{C}_\varphi)$, we have that $((p_{i_\kappa}^\kappa), \psi), w_{i_\kappa}^\kappa) \in L(\mathcal{C}_\varphi)$.

Induction step: Assume that $((p_i^\kappa, \psi), w_i^\kappa) \in L(\mathcal{C}_\varphi)$ for all $\kappa \in \rho$ and $0 < i \leq i_\kappa$. Consider an arbitrary $\kappa \in \rho$ and $((p_{i-1}^\kappa, \psi), w_{i-1}^\kappa)$. Then, there exists a transition $(p_{i-1}^\kappa, w_{i-1}^\kappa) \Delta X$ such that $X = \{(p_i^{\kappa'}, w_i^{\kappa'}) \mid \kappa' \in \rho, (p_{i-1}^{\kappa'}, w_{i-1}^{\kappa'}) = (p_{i-1}^\kappa, w_{i-1}^\kappa)\}$; i.e., the transition taken at $(p_{i-1}^\kappa, w_{i-1}^\kappa)$ in ρ. Then, by induction hypothesis, $((p_i^{\kappa'}, \psi), w_i^{\kappa'}) \in L(\mathcal{C}_\varphi)$ for all $\kappa' \in \rho$ with $(p_{i-1}^{\kappa'}, w_{i-1}^{\kappa'}) = (p_{i-1}^\kappa, w_{i-1}^\kappa)$; i.e., $((p', \psi), w') \in L(\mathcal{C}_\varphi)$ for all $(p', w') \in X$. Moreover, we have that (i) $((p_{i-1}^\kappa, \psi), w_{i-1}^\kappa) \Rightarrow_{\mathcal{C}_\varphi} \{((p_{i-1}^\kappa, \psi_1), w_{i-1}^\kappa)\} \cup \{((p', \psi), w') \mid (p', w') \in X\}$ and (ii) $((p_{i-1}^\kappa, \psi_1), w_{i-1}^\kappa) \in L(\mathcal{C}_\varphi)$. Therefore, $((p_{i-1}^\kappa, \psi), w_{i-1}^\kappa) \in L(\mathcal{C}_\varphi)$.

For the other cases of ψ, the proofs are similar.

$(\Rightarrow)$: Assume that $((p, \psi), w) \in L(\mathcal{C}_\varphi)$, then we prove that $\mathcal{C}, (p, w), \mathsf{lab} \models \psi$ by induction on the structure of ψ.

Case $\psi = \mathsf{p}$**:** Since $((p, \mathsf{p}), w) \in L(\mathcal{C}_\varphi)$, there exists an accepting $(p, \mathsf{p}), w)$-run ρ. Furthermore, the prefix of ρ must satisfy the following: $((p, \mathsf{p}), w) \Rightarrow_{\mathcal{C}_\varphi} (p_\mathsf{p}, w) \Rightarrow^*_{\mathcal{C}_\varphi} (f, \#)$ for some $f \in F_\mathsf{p}$. Thus, $\mathcal{A}_\mathsf{p}$ has a run $p_\mathsf{p} \xrightarrow{w}^*_{\mathcal{A}_\mathsf{p}} f$, i.e., $(p, w) \in L(\mathcal{A}_\mathsf{p}) = \mathsf{lab}(\mathsf{p})$. Hence, $\mathcal{C}, (p, w), \mathsf{lab} \models \mathsf{p}$

Case $\psi = \psi_1 \vee \psi_2$**:** Since $((p, \psi_1 \vee \psi_2), w) \in L(\mathcal{C}_\varphi)$, there exists an accepting $((p, \psi_1 \vee \psi_2), w)$-run ρ. Furthermore, ρ must have the form $((p, \psi_1 \vee \psi_2), w)(\rho')$ where ρ' is rooted at either $((p, \psi_1), w)$ or $((p, \psi_2), w)$. Without loss of generality, let us assume $((p, \psi_1), w)$ is the root of ρ'; then ρ' is an accepting run, i.e., $((p, \psi_1), w) \in L(\mathcal{C}_\varphi)$ By induction hypothesis, we have $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1$; thus, $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1 \vee \psi_2$.

Case $\psi = \psi_1 \wedge \psi_2$**:** Since $((p, \psi_1 \wedge \psi_2), w) \in L(\mathcal{C}_\varphi)$, there exists an accepting $((p, \psi_1 \wedge \psi_2), w)$-run ρ. Furthermore, ρ must have the form $((p, \psi_1 \wedge \psi_2), w)(\rho_1, \rho_2)$ where ρ_i is rooted at $((p, \psi_i), w)$. Then, ρ_1 and ρ_2 are both accepting, i.e., $((p, \psi_1), w) \in L(\mathcal{C}_\varphi)$ and $((p, \psi_2), w) \in L(\mathcal{C}_\varphi)$. By induction hypothesis, we have $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1$ and $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_2$; thus, $\mathcal{C}, (p, w), \mathsf{lab} \models \psi_1 \wedge \psi_2$.

Case $\psi = \mathsf{EX}\psi_1$**:** Since $((p, \mathsf{EX}\psi_1), w) \in L(\mathcal{C}_\varphi)$, there exists an accepting $((p, \mathsf{EX}\psi_1), w)$-run ρ. Furthermore, ρ must have the form $((p, \mathsf{EX}\psi_1), w)(\rho_1, \ldots, \rho_n)$ for some $(p, a_1 \ldots a_m) \Delta \{(p_1, w_1), \ldots, (p_n, w_n)\}$ where $w = w' a_1 \ldots a_m$ and ρ_i is rooted at $((p_i, \psi_1), w' w_i')$. Then, for all $1 \leq i \leq n$, ρ_i is accepting, i.e., $((p_i, \psi_1), w' w_i') \in L(\mathcal{C}_\varphi)$. By induction hypothesis, we have $\mathcal{C}, (p_i, w' w_i'), \mathsf{lab} \models \psi_1$; thus, $\mathcal{C}, (p, w), \mathsf{lab} \models \mathsf{EX}\psi_1$.

Case $\psi = \mathsf{E}\psi_1 \mathbf{U} \psi_2$**:** Since $((p, \mathsf{E}\psi_1 \mathbf{U} \psi_2), w) \in L(\mathcal{C}_\varphi)$, there exists an accepting $((p, \mathsf{E}\psi_1 \mathbf{U} \psi_2), w)$-run ρ. We convert ρ into a prefix $g(\rho)$ of some run in $\mathcal{C}$ by induction on the structure of ρ as follows:

- $g(((p, \mathsf{E}\psi_1 \mathbf{U} \psi_2), w)(\rho')) = (p, w)$ where ρ' is the only direct sub-tree of the root of ρ according to Rule 7 given in Section 3.3; and

– $g(((p, \mathsf{E}\psi_1\mathbf{U}\psi_2), w)(\rho', \rho_1, \ldots, \rho_n)) = (p, w)(g(\rho_1), \ldots, g(\rho_n))$ for some $((p, \mathsf{E}\psi_1\mathbf{U}\psi_2), w) \Rightarrow_{\mathcal{C}_\varphi} \{((p, \psi_1), w)\} \cup \{((p, \mathsf{E}\psi_1\mathbf{U}\psi_2), w') \mid (p', w') \in X\}$ where $(p, w) \Rightarrow_{\mathcal{C}} X$ according to Rule 8 given in Section 3.3.

Then, every path $\kappa = (p_0, w_0) \ldots (p_m, w_m) \in g(\rho)$ (for some $m \geq 0$, $p_0 = p$, and $w_0 = w$) corresponds to a prefix of a path in ρ which has the form $((p_0, \mathsf{E}\psi_1\mathbf{U}\psi_2), w_0)$ $\ldots ((p_m, \mathsf{E}\psi_1\mathbf{U}\psi_2), w_m)((p_m, \psi_2), w_m)$ for some $m \geq 0$. Furthermore, for all $i < m$, $((p_i, \psi_1), w_i)$ is the direct child of $((p_i, \mathsf{E}\psi_1\mathbf{U}\psi_2), w_i)$. Then, for all $i < m$, $((p_i, \psi_1),$ $w_i) \in L(\mathcal{C}_\varphi)$ and $((p_m, \psi_2), w_m) \in L(\mathcal{C}_\varphi)$. By induction hypothesis, we have $\mathcal{C}, (p_i, w_i$ $), \mathsf{lab} \models \psi_1$ for all $i < m$ and $\mathcal{C}, (p_m, w_m), \mathsf{lab} \models \psi_2$; thus $\mathcal{C}, (p, w), \mathsf{lab} \models \mathsf{E}\psi_1\mathbf{U}\psi_2$.

For the other cases of ψ, the proofs are similar. □

The following theorem follows from this Lemma 2, Proposition 1, and Theorem 1.

Theorem 4. *For a given CABPDS $\mathcal{C}$, a regular labelling function* lab*, and a* CTL*-formula φ there is an effectively computable alternating automaton $\mathcal{A}_{\mathcal{C},\varphi}$ such that for all configurations $c = (p, w) \in \mathsf{Cnf}_{\mathcal{C}}$ the following holds: $\mathcal{C}, c, \mathsf{lab} \models \varphi$ iff $((p, \varphi), w) \in L(\mathcal{A}_{\mathcal{C},\varphi})$.*

Proof. Let $\mathcal{C}$ be an CABPDS, φ a CTL-formula, (p, w) a configuration, and lab a regular labelling function. We construct the CABPDS $\mathcal{C}_\varphi$ with $(\star)$ $\mathcal{C}, (p, w) \models \varphi$ iff $((p, \varphi), w) \in L(\mathcal{C}_\varphi)$ according to Lemma 2. We apply Proposition 1 to obtain: $(\star)$ iff $((p, \varphi), w) \in L(\mathcal{B}(\mathcal{C}_\varphi))$ where $\mathcal{B}(\mathcal{C}_\varphi)$ is an ABPDS. Finally, by Theorem 1 we can conclude that there is an effectively constructable alternating $\mathcal{B}(\mathcal{C}_\varphi)$-automaton $\mathcal{A}$ with $(\star)$ iff $((p, \varphi), w) \in L(\mathcal{A})$. □

4 Decidability of RAL over 1-Unbounded Models

Throughout this section we assume that $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, , \mathfrak{R}, t)$ is an 1-unbounded $\mathbf{RBM}$ where $\mathfrak{R}$ is a shared resource structure consisting of a single unbounded shared resource. Moreover, let A be a set of agents and $\bar{A} = \mathbb{A}\mathrm{gt} \backslash A$. As there is only one resource, we can simplify the notation. We write η for $\eta(r)$, $\mathsf{cons}(\alpha)$ instead of $\mathsf{cons}(\alpha, r)$ and so on. Also, for an action profile $\boldsymbol{\alpha}_A$ we use $\mathsf{cons}(\boldsymbol{\alpha}_A)$ (resp. $\mathsf{prod}(\boldsymbol{\alpha}_A)$) to refer to $\sum_{a \in A} \mathsf{cons}(\alpha_a)$ (resp. $\sum_{a \in A} \mathsf{prod}(\alpha_a)$). Furthermore, for a natural number x, $\mathbf{[}x\mathbf{]_1}$ is used to refer to a sequence $|| \ldots |$ of x lines each representing one element on the stack, i.e. $\mathbf{[}x\mathbf{]_1}$ corresponds to the *unary encoding* of x. We write $\mathbf{[}0\mathbf{]_1} = \epsilon$. Similarly, we use $\mathbf{[}y\mathbf{]_{10}}$ to refer to the *ternary encoding* of $y = \mathbf{[}x\mathbf{]_1}$ for a natural number x.

4.1 Encoding of an iRBM

We define the following auxiliary functions where q is a state in $\mathfrak{M}$, $\boldsymbol{\alpha}_A$ a joint action of A and $\boldsymbol{\alpha}_{\bar{A}}$ a joint action of $\bar{A}$:

$$\Delta\mathsf{max}_{\bar{A}}(q) = \max\{\mathsf{cons}(\boldsymbol{\alpha}_{\bar{A}}) \mid \boldsymbol{\alpha}_{\bar{A}} \in d_{\bar{A}}(q)\}$$
$$\Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A) = \mathsf{cons}(\boldsymbol{\alpha}_A) + \Delta\mathsf{max}_{\bar{A}}(q)$$
$$\Delta\mathsf{prd}_A(q, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}}) = \Delta\mathsf{max}_{\bar{A}}(q) - \mathsf{cons}(\boldsymbol{\alpha}_{\bar{A}}) + \mathsf{prod}((\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}}))$$

The number $\Delta\mathsf{max}_{\bar{A}}(q)$ denotes the worst case consumption of resources of the opponents at q, that is the maximal amount of resources they could claim. The number $\Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A)$ is the consumption of resources if A executes $\boldsymbol{\alpha}_A$ and the opponents choose their actions with the worst case consumption; this models a pessimistic view. This is valid as the proponents can never be sure to have more resources available. Finally, $\Delta\mathsf{prd}_A(q, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})$ denotes the number of resources that need to be produced after $(\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})$ was executed at q. It is the sum of the number of resources produced by $(\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})$, and the difference between the consumption of the estimated worst case behavior of the opponents and the consumption of the actions which were actually executed by $\bar{A}$. We state the following lemma which is fundamental for the correctness of the encoding defined below. It justifies that we can first assume the worst-case behavior of the opponents before correcting this choice.

Lemma 3. *Let $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})$ be a tuple consisting of an action profile $\boldsymbol{\alpha}_A$ of A, $\boldsymbol{\alpha}_{\bar{A}}$ be one of $\bar{A}$ and q be a state in $\mathfrak{M}$. We have that*

(a) $\mathsf{prod}(\boldsymbol{\alpha}) - \mathsf{cons}(\boldsymbol{\alpha}) = \Delta\mathsf{prd}_A(q, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}}) - \Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A)$*; and*
(b) the following statements are equivalent for any natural number x:
  (i) for all $\boldsymbol{\alpha}' \in Act_{\bar{A}}$: $x \geq \sum_{a\in\bar{A}} \mathsf{cons}(\alpha'_a) + \sum_{a\in A} \mathsf{cons}(\alpha_a)$.
  (ii) $x \geq \Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A)$.

Proof. (a) $\Delta\mathsf{prd}_A(q, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}}) - \Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A) = \Delta\mathsf{max}_{\bar{A}}(q) - \mathsf{cons}(\boldsymbol{\alpha}_{\bar{A}}) + \mathsf{prod}((\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})) - (\mathsf{cons}(\boldsymbol{\alpha}_A) + \Delta\mathsf{max}_{\bar{A}}(q)) = \mathsf{prod}((\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})) - (\mathsf{cons}(\boldsymbol{\alpha}_A) + \mathsf{cons}(\boldsymbol{\alpha}_{\bar{A}})) = \mathsf{prod}(\boldsymbol{\alpha}) - \mathsf{cons}(\boldsymbol{\alpha})$.
(b) $\sum_{a\in\bar{A}} \mathsf{cons}(\alpha'_a) + \sum_{a\in A} \mathsf{cons}(\alpha_a) \leq x$ for all $\boldsymbol{\alpha}' \in Act_{\bar{A}}$ iff $\mathsf{cons}(\boldsymbol{\alpha}_A) + \Delta\mathsf{max}_{\bar{A}}(q) \leq x$ iff $\Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A) \leq x$.

□

From $\mathfrak{M}$ and A, we define an r-compact ABPDS where $r = \mathbf{[}\max_{q,\boldsymbol{\alpha}_A,\boldsymbol{\alpha}_{\bar{A}}}\{\Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A), \Delta\mathsf{prd}_A(q, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})\}\mathbf{]_1}$ is the maximal number which is ever *consumed* or *produced*.

Definition 3 **($\mathcal{C}_{\mathfrak{M},A}$).** *The r-compact ABPDS $\mathcal{C}_{\mathfrak{M},A}$ is the CABPDS $(S, \Gamma, \Delta, F, r)$ where $S = F = Q$, $\Gamma = \{|\}$, and for all $q \in Q$, $\boldsymbol{\alpha}_A \in d_A(q)$ we have that*

$$(q, \mathbf{[}\Delta\mathsf{con}_A(q, \boldsymbol{\alpha}_A)\mathbf{]_1}) \Delta \{(o(q, (\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})), \mathbf{[}\Delta\mathsf{prd}_A(q, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})\mathbf{]_1}) \mid \boldsymbol{\alpha}_{\bar{A}} \in d_{\bar{A}}(q)\}.$$

It is easily seen that $\mathcal{C}_{\mathfrak{M},A}$ is indeed an r-compact ABPDS. The purpose of $\mathcal{C}_{\mathfrak{M},A}$ is to encode the outcome sets $out(q, s_A, \eta)$ for any state q and strategy s_A. Let $w \in \{|\}^*$ and $\rho \in \mathcal{R}_{\mathcal{C}_{\mathfrak{M},A}}$. We define $h(w) = \mathbf{[}w\mathbf{]_{10}}$ and lift h to configurations $h((p, w)) = (p, h(w))$, to finite or infinite sequences $c_0c_1 \ldots$ of configurations via $h(c_0c_1 \ldots) = h(c_0)h(c_1) \ldots$, and to runs $h(\rho) = \{h(\kappa) \mid \kappa \in \rho\}$. Then, the next result states that runs of $\mathcal{C}_{\mathfrak{M},A}$ are the outcome sets of A. First, observe that for every strategy s_A we have that there is a run $\rho \in \mathcal{R}_{\mathcal{C}_{\mathfrak{M},A}}$ with $h(\rho) = out(q, s_A, \eta)$. The automaton simply chooses the same actions as specified by the strategy. Similarly, in the reverse case, if the automaton takes a transition corersponding to an action tuple $\boldsymbol{\alpha}_A$ after the finite run b, then we define the strategy s_A such that $s_A(h(b)) = \boldsymbol{\alpha}_A$. We note that here it is important that the strategy is perfect recall and takes the hisotry of states as well as of shared endowments into account.

Lemma 4 (Encoding Lemma). *$h : \mathcal{R}_{\mathcal{C}_{\mathfrak{M},A}} \to \{out(q, s_A, \eta) \mid (q, \eta) \in Q \times \mathsf{En}$ and s_A is a strategy of $A\}$ is an isomorphism.*

Proof (Sketch). The proof is done by induction on the number of simulation steps. Let s_A be a strategy and $t = out(q, s_A, \eta)$ the (q, s_A, η)-outcome. First, we argue that there is a run $\rho \in \mathcal{R}_{\mathcal{C}_{\mathfrak{M},A}}((q, [\![\eta]\!]_{\mathbf{1}}))$ with $h(\rho) = t$. Let t^i and ρ^i be the finite version of t and ρ up to depth $i \geq 0$, respectively. We construct ρ step-by-step. Clearly, $h(\rho^0) = h(t^0)$. Let b^i be any finite branch in t^i with final configuration (q', η') and with successor states $\{(q_1, \eta_1), \ldots, (q_n, \eta_n)\}$ and let the action $\boldsymbol{\alpha}^j_{\bar{A}}$ of the opponents be the action which led to (q_j, η_j) given that $s_A(t^i) = \boldsymbol{\alpha}_A$, i.e. $o(q', (\boldsymbol{\alpha}_A, \boldsymbol{\alpha}^j_{\bar{A}})) = q_j$, for $1 \leq j \leq n$. By definition there is a transition $(q', [\![\Delta\mathsf{con}_A(q', \boldsymbol{\alpha}_A)]\!]_{\mathbf{1}}) \Delta \{(q_j, [\![\Delta\mathsf{prd}_A(q', \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})]\!]_{\mathbf{1}}) \mid \boldsymbol{\alpha}_{\bar{A}} \in 1 \leq j \leq n\}$. Let κ^i be the finite branch on ρ^i corresponding to b^i, by induction we have $h(\kappa^i) = b^i$. The last state on κ^i is $c' = (q', [\![\eta']\!]_{\mathbf{1}})$. By Lemma 3(b) and the fact that there is a transition after b^i, $\eta' \geq \Delta\mathsf{con}_A(q', \boldsymbol{\alpha}_A)$. Thus, the transition of the automaton can be taken and by Lemma 3(a), $\{(q_1, [\![\eta_1]\!]_{\mathbf{1}}), \ldots, (q_n, [\![\eta_n]\!]_{\mathbf{1}})\}$ is a direct successor of c'.

For the other direction, let $\rho \in \mathcal{R}_{\mathcal{C}_{\mathfrak{M},A}}(c)$. The proof is done in a similar way. Let $\kappa^i = (q_0, w_0) \ldots (q_n, w_n)$ be a finite branch in ρ and assume that the automaton takes as next transition $(q_n, [\![\Delta\mathsf{con}_A(q_n, \boldsymbol{\alpha}_A)]\!]_{\mathbf{1}}) \Delta \{(o(q_n, (\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})), [\![\Delta\mathsf{prd}_A(q_n, \boldsymbol{\alpha}_A, \boldsymbol{\alpha}_{\bar{A}})]\!]_{\mathbf{1}}) \mid \boldsymbol{\alpha}_{\bar{A}} \in d_{\bar{A}}(q)\}$. Then, we define $s_A(h(\kappa^i)) = \boldsymbol{\alpha}_A$. To see that s_A is well-defined we observe that ρ cannot contain two finite banches b and b' which are identical. □

4.2 Model Checking RAL over 1-Unbounded RBMs is Decidable

In this section we put the pieces together and show that model checking RAL over 1-unbounded **RBM**s is decidable. Before we do so, we need to extend **RBM**s with regular labelling functions $\pi : \Pi \to 2^{Q \times \mathsf{En}}$ as done in Section 3.3 for PDSs. Clearly, the "sate-based" labelling function $\pi' : \Pi \to 2^Q$ in $\mathfrak{M}$ is a special regular labelling function with $\pi(\mathsf{p}) = \{(q, \eta) \mid \eta \in \mathsf{En}, q \in \pi'(\mathsf{p})\}$. From now on, we assume that π is regular. Our model checking algorithm builds upon model checking CTL formulae over CABPDSs as outlined in Theorem 4. The main idea is the following. Suppose we want to model check $\mathfrak{M}, q_0, \eta \models \langle\langle A \rangle\rangle^{\downarrow} \varphi$ where $\langle\langle A \rangle\rangle^{\downarrow} \varphi$ is a flat formula and $\mathsf{E}\varphi$ is in negation normal form [9]. Firstly, we construct the CABPDS $\mathcal{C}_{\mathfrak{M},A}$ which accepts the outcome sets of A by the Encoding Lemma 4. Let lab be the labelling function defined as: $(q, [\![\eta]\!]_{\mathbf{1}}) \in \mathsf{lab}(\mathsf{p})$ iff $(q, \eta) \in \pi(\mathsf{p})$. Then, we have that: $\mathfrak{M}, q_0, \eta \models \langle\langle A \rangle\rangle^{\downarrow} \varphi$ if, and only if, $\mathcal{C}_{\mathfrak{M},A}, (q, [\![\eta]\!]_{\mathbf{1}}), \mathsf{lab} \models \mathsf{E}\varphi$. By Theorem 4 this can be efficiently solved by constructing an alternating automaton $\mathcal{A}_{\mathcal{C}_{\mathfrak{M},A}, \mathsf{E}\varphi}$ that accepts $((q, \mathsf{E}\varphi), [\![\eta]\!]_{\mathbf{1}})$ iff the above equivalence is true. This shows the following result:

Proposition 2. *Let the labelling function in $\mathfrak{M}$ be regular and $\langle\langle A \rangle\rangle \varphi$ be a flat* RAL*-formula in negation normal form. Then, we can construct an alternating automaton $\mathcal{A}_{\mathcal{C}_{\mathfrak{M},A}, \mathsf{E}\varphi}$ such that $((q, \mathsf{E}\varphi), [\![\eta]\!]_{\mathbf{1}}) \in L(\mathcal{A}_{\mathcal{C}_{\mathfrak{M},A}, \mathsf{E}\varphi})$ if, and only if, $\mathfrak{M}, q, \eta \models \langle\langle A \rangle\rangle^{\downarrow} \varphi$.*

[9] Note that the release operator cannot occur here.

This proposition can be applied recursively to model check an arbitrary RAL-formula φ, following the standard bottom-up model checking approach used for $\mathbf{CTL}^*$ [11]. Firstly, the innermost (flat) formulae ψ of φ are considered. By Proposition 2 we can compute the regular set of configurations at which each of these subformulae ψ hold. Then, we replace the subformula ψ by a fresh propositions p_ψ and extend the regular labelling of $\mathfrak{M}$ such that p_ψ is assigned the configurations at which ψ is true (Theorem 4). Applied recursively, we obtain:

Theorem 5. *The model-checking problem for* RAL *(with shared resources) over* 1*-unbounded* $\mathbf{RBM}$*s is decidable.*

Proof (Sketch). The proof proceeds by induction on the formula structure. Suppose we want to model check $\mathfrak{M}, q, \eta \models \langle\!\langle A \rangle\!\rangle \mathbf{F}\varphi$. The other cases are handled analogously. Let $\xi = \langle\!\langle B \rangle\!\rangle \chi$ be any strict subformula of φ. By induction hypothesis and Lemma 2 we can construct an alternating automaton $\mathcal{A}_{\mathcal{C}_{\mathfrak{M},B},\mathsf{E}\chi}$ that accepts exactly those $((q', \mathsf{E}\chi), h(\eta'))$ with $\mathfrak{M}, q', \eta' \models \xi$. Then, we replace ξ in φ with a fresh proposition p_ξ and extend π by defining $\pi(p_\xi) = L(\mathcal{A}'_{\mathcal{C}_{\mathfrak{M},B},\mathsf{E}\chi})$ where $\mathcal{A}'_{\mathcal{C}_{\mathfrak{M},B},\mathsf{E}\chi}$ is the automaton with $L(\mathcal{A}'_{\mathcal{C}_{\mathfrak{M},B},\mathsf{E}\chi}) = \{(q, \eta) \mid ((q, \mathsf{E}\chi), [\eta]_{\mathbf{1}}) \in L(\mathcal{A}_{\mathcal{C}_{\mathfrak{M},B},\mathsf{E}\chi})\}$. We proceed with this procedure until the "updated" φ is completely propositional. Then, we can apply Proposition 2 to check whether $\mathfrak{M}, q, \eta \models \langle\!\langle A \rangle\!\rangle \mathbf{F}\varphi$. □

5 General Undecidability Result

In [3,8] it has been shown that most variants of RAL are undecidable. This has been proved by reductions of the halting problem of two-counter automata [14] to the different model checking problems. Two counter automata are finite automata extended with two counters. Transitions depend on the current state of the automaton and on whether the counters are zero or non-zero. If a transition is taken, the automaton may change its control state and may increment or decrement the counters. The basic idea of the reduction is to encode a two counter automaton as an $\mathbf{iRBM}$[10]. Each of the two counters corresponds to a resource type. Agents' actions are used to simulate the selection of a transition and the incrementation and decrementation of counters. The key difficulty is to encode the *zero test*, i.e. to check whether resources are available. The two counter automaton can check if a counter is zero or not in the transition relation by definition. But, if in the resource bounded model a transition should only be taken if no resources are available, there is nothing which can prevent the agent to take the transition even if it has resources available. Clearly, such an inconsistent behavior would break the simulation. Therefore, a second agent, playing the role of a spoiler, is used to check that such inconsistent transitions result in a "fail states" which cannot be used to witness an accepting run of the automaton. Then, it is shown that the two-counter automaton halts on the empty input iff $\langle\!\langle 1 \rangle\!\rangle^{\downarrow} \mathbf{F}\mathsf{halt}$ is true in a model which encodes the transition table of the automaton [3]. In another result the authors of [3] also show that undecidability is the case for a single agent only. This is achieved by nesting modalities and letting the

[10] We note that we show undecidability over $\mathbf{iRBM}$s. Such undecidability result are stronger than for $\mathbf{RBM}$s as the former is a special case of the latter.

agent itself play the role of the spoiler: the two-counter automaton halts on the empty input iff $\langle\langle 1\rangle\rangle^{\downarrow}(\neg\langle\langle 1\rangle\rangle^{\downarrow}\mathbf{X}\mathsf{err})\mathbf{U}\mathsf{halt}$. These undecidability proofs can be (directly) adapted to our setting; actually, due to the shared resources the technicalities are even simpler. We note that the undecidability proof does not require the full expressivity of strategies as dedined in this paper. Strategies which only take the history of states into account are sufficient to encode the behavior of a two-counter automaton. This corresponds to the fact that the automaton takes transitions based on the control states and whether the counters are zero or non-zero, but not the actual counter value. We refer to [3,8] for further details about the construction. We obtain the following result:

Corollary 2 (of [8,3]). *Model checking* RAL *(with shared resoures) over k-unbounded* $\mathbf{iRBM}s$ *with $k \geq 2$ is undecidable, even in the following restricted cases:*

1. *In the case of a single agent and a fixed formula of the form* $\langle\langle 1\rangle\rangle^{\downarrow}(\neg\langle\langle 1\rangle\rangle^{\downarrow}\boldsymbol{X}\mathsf{p})\boldsymbol{U}\mathsf{q}$.
2. *In the case of two agents and a fixed formula of the form* $\langle\langle 1\rangle\rangle^{\downarrow}\mathbf{F}\mathsf{p}$.

6 Conclusions

In this paper, we have introduced a variant of resource agent logic RAL [8] with shared resources, which can be consumed and produced. We showed that the model checking problem is undecidable in the presence of at least two unbounded resource types. Our main technical result is a decidability proof of model checking RAL with one shared, unbounded resource type. Otherwise, we impose no restrictions, in particular nested cooperation modalities do not reset the resources available to agents. This property is sometimes called *non-resource flatness*. In order to show decidability, we first show how CTL can be model-checked with respect to (compact) alternating Büchi pushdown systems extending results on model checking CTL over pushdown and alternating pushdown systems [15,5]. A compact alternating Büchi pushdown system allows to read and to pop more than one symbol from its stack at a time. It is used for encoding resource bounded models in order to apply the automata-based model checking algorithm.

Acknowledgement. We would like to thank Natasha Alechina and Brian Logan for the many discussions on this topic and their valuable comments.

References

1. N. Alechina, B. Logan, H. N. Nguyen, and F. Raimondi. Decidable model-checking for a resource logic with production of resources. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-2014)*, pages 9–14. ECCAI, IOS Press, 2014.
2. N. Alechina, B. Logan, H. N. Nguyen, and A. Rakib. Resource-bounded alternating-time temporal logic. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 481–488. IFAAMAS, 2010.
3. Natasha Alechina, Nils Bulling, Brian Logan, and Hoang Nga Nguyen. On the boundary of (un)decidability: Decidable model-checking for a fragment of resource agent logic. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1494–1501, 2015.
4. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

5. Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR'97: Concurrency Theory*, pages 135–150. Springer, 1997.
6. Laura Bozzelli. Complexity results on branching-time pushdown model checking. *Theoretical computer science*, 379(1):286–297, 2007.
7. N. Bulling and B. Farwer. Expressing properties of resource-bounded systems: The logics RBTL and RBTL*. In *Post-Proceedings of CLIMA '09*, number 6214 in LNCS 6214, pages 22–45, 2010.
8. N. Bulling and B. Farwer. On the (un-)decidability of model checking resource-bounded agents. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215, pages 567–572. IOS Press, 2010.
9. Nils Bulling and Hoang Nga Nguyen. Model checking resource bounded systems with shared resources via alternating Büchi pushdown systems (to appear). In *Proc. of the 18th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2015)*, Bertinoro, Italy, October 2015.
10. Thierry Cachat. Symbolic strategy synthesis for games on pushdown graphs. In *Automata, Languages and Programming*, pages 704–715. Springer, 2002.
11. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
12. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.
13. D. Della Monica, M. Napoli, and M. Parente. Model checking coalitional games in shortage resource scenarios. In *Proceedings of the 4th International Symposium on Games, Automata, Logics and Formal Verification (GandALF 2013*, volume 119 of *EPTCS*, pages 240–255, 2013.
14. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
15. Fu Song and Tayssir Touili. Efficient CTL model-checking for pushdown systems. *Theoretical Computer Science*, 549:127–145, 2014.
16. Dejvuth Suwimonteerabuth, Stefan Schwoon, and Javier Esparza. Efficient algorithms for alternating pushdown systems: Application to certificate chain discovery with threshold subjects. 2006.